**Introduction to various translators**

**Assembler, Compilers and Interpreters**

Any program that is not written in machine language has to be translated in machine language before it is executed by the computer. The means used for translation are themselves computer programs. There are three types of translator programs i.e. Assembler, Compilers and Interpreters.

**Assembler:**

Assembler is a computer program which is used to translate program written in Assembly Language in to machine language. The translated program is called as object program. Assembler checks each instruction for its correctness and generates diagnostic messages, if there are mistakes in the program.

Various steps of assembling are:

1. Input source program in Assembly Language through an input device
2. Use Assembler to produce object program in machine language.
3. Execute the program.

**Compiler:**

A compiler is a program that translates a programme written in HLL to executable machine language. The process of transferring HKK source program in to object code is a lengthy and complex process as compared to assembling.

Compliers have diagnostic capabilities and prompt the programmer with appropriate error message while compiling a HLL program. The corrections are to be incorporated in the program, whenever needed, and the program has to be recompiled. The process is repeated until the program is mistake free and translated to an object code.

Thus the job of a complier includes the following:

1. To translate HLL source program to machine codes.

2. To trace variables in the program

3. To include linkage for subroutines.

4. To allocate memory for storage of program and variables.

5. To generate error messages, if there are errors in the program.

**Interpreter:**

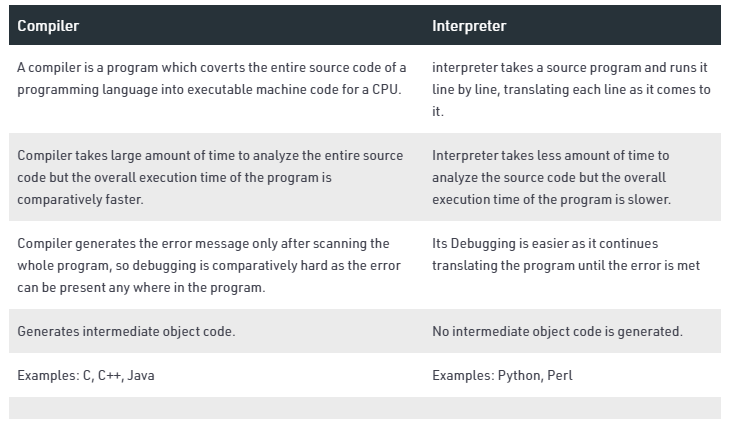
The basic purpose of interpreter is same as that of complier. In compiler, the program is translated completely and directly executable version is generated. Whereas interpreter translates each instruction, executes it and then the next instruction is translated and this goes on until end of the program. In this case, object code is not stored and reused.

Every time the program is executed, the interpreter translates each instruction freshly. It also has program diagnostic capabilities. However, it has some disadvantages as below:

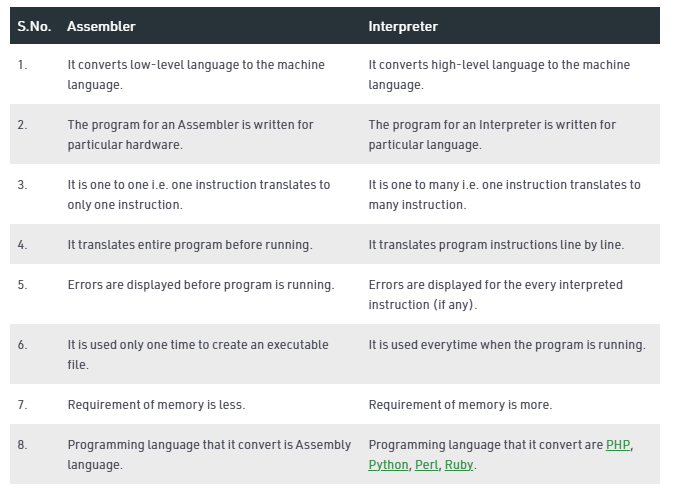
1. Instructions repeated in program must be translated each time they are executed.

2. Because the source program is translated fresh every time it is used, it is slow process or execution takes more time. Approx. 20 times slower than complier.

Compiler Vs Interpreter



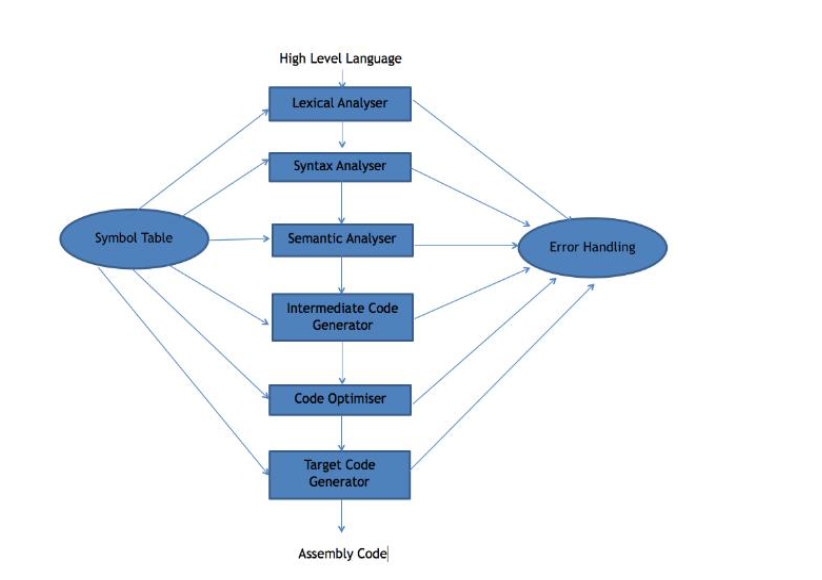
Assembler Vs Interpreter



**Compiler:**

A compiler is a program that translates a programme written in HLL to executable machine language. The process of transferring HKK source program in to object code is a lengthy and complex process as compared to assembling.

**Phases of compiler**



**Symbol Table –** It is a data structure being used and maintained by the compiler, consists all the identifier’s name along with their types. It helps the compiler to function smoothly by finding the identifiers quickly.

The compiler has two modules namely front end and back end. Front-end constitutes of the Lexical analyzer, semantic analyzer, syntax analyzer and intermediate code generator. And the rest are assembled to form the back end.

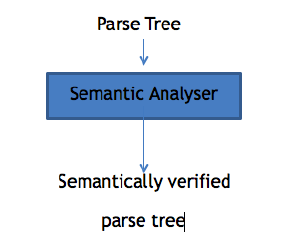
1. **Lexical Analyzer –** It reads the program and converts it into tokens. It converts a stream of lexemes into a stream of tokens. Tokens are defined by regular expressions which are understood by the lexical analyzer. It also removes white-spaces and comments.

2. **Syntax Analyzer –** It is sometimes called as parser. It constructs the parse tree. It takes all the tokens one by one and uses Context Free Grammar to construct the parse tree.

**Why Grammar ?**

The rules of programming can be entirely represented in some few productions. Using these productions we can represent what the program actually is. The input has to be checked whether it is in the desired format or not.

Syntax error can be detected at this level if the input is not in accordance with the grammar.



**3.Semantic Analyzer –** It verifies the parse tree, whether it’s meaningful or not. It furthermore produces a verified parse tree.

**4.Intermediate Code Generator –** It generates intermediate code, that is a form which can be readily executed by machine We have many popular intermediate codes. Example – Three address code etc. Intermediate code is converted to machine language using the last two phases which are platform dependent.

Till intermediate code, it is same for every compiler out there, but after that, it depends on the platform. To build a new compiler we don’t need to build it from scratch. We can take the intermediate code from the already existing compiler and build the last two parts.

**5.Code Optimizer –** It transforms the code so that it consumes fewer resources and produces more speed. The meaning of the code being transformed is not altered. Optimization can be categorized into two types: machine dependent and machine independent.

**6.Target Code Generator –** The main purpose of Target Code generator is to write a code that the machine can understand. The output is dependent on the type of assembler. This is the final stage of compilation.

Online Notes

<https://ce101ms.files.wordpress.com/2014/02/compiler_assembler_linker_loader.pdf>

<http://ggn.dronacharya.info/APSDept/Downloads/QuestionBank/Fundamentals-Computer-Programming-C/Section-B/Lecture8.pdf>

<https://dspace.mit.edu/bitstream/handle/1721.1/56573/6-004Fall-2002/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-004Computation-StructuresFall2002/4F687FF5-CD7E-465C-A5AF-9C15ECD9589D/0/L13MachineLanguage.pdf>

Video Links

<https://www.youtube.com/watch?v=dyDZO-4R1Nw>

<https://www.youtube.com/watch?v=qcEBXrqFLe8>

<https://www.youtube.com/watch?v=hD0db1VkVoU>